BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 December 2001 (13.12.2001)

PCT

(10) International Publication Number
WO 01/95104 A1

(51) International Patent Classification[7]: G06F 9/45, 9/445

(21) International Application Number: PCT/US01/16993

(22) International Filing Date: 24 May 2001 (24.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/587,075 2 June 2000 (02.06.2000) US

(71) Applicant: YAHOO INC. [US/US]; 701 First Avenue, Sunnyvale, CA 94089 (US).

(72) Inventor: SHAFRON, Thomas, J.; Apt. #1601, 1483 Sutter Street, San Francisco, CA 94109 (US).

(74) Agents: DECARLO, James, J. et al.; Stroock & Stroock & Lavan LLP, 180 Maiden Lane, New York, NY 10038 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

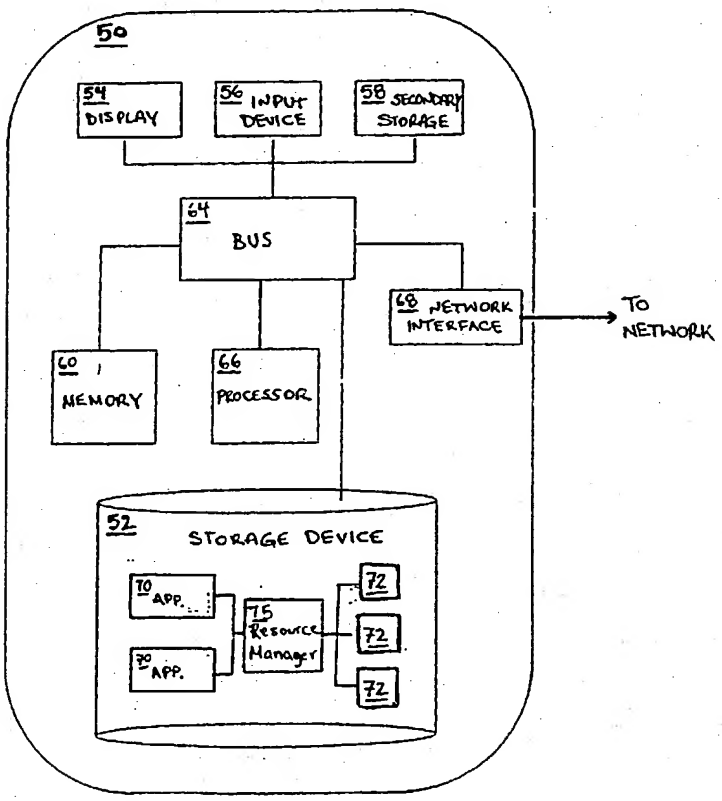
Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for all designations*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

[Continued on next page]

(54) Title: A METHOD AND SYSTEM FOR MANAGING THE RESOURCES OF AN APPLICATION PROGRAM



(57) Abstract: A software resource manager (75) is stored on a client computer (50) and maintains a map of unique identifiers to resources, so as to determine those resources that are stored locally on the client computer and those resources that must be retrieved from a server system. The resource manager is further programmed to either pass the resources to one or more software applications (70) or respond to a function call from the software application indicating that resources are needed. In this way, one or more software applications can be internationalized or customized without the need to distribute all of the resources for the application and without the need to distribute all of the resources for the application and without the need to restart the application.

WO 01/95104 A1

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A METHOD AND SYSTEM FOR MANAGING THE RESOURCES OF AN APPLICATION PROGRAM

FIELD OF THE INVENTION

The present invention is directed to the field of software systems and applications, and, more particularly, to a method and system for dynamically managing the resources of at least one application stored on a computer and for permitting the customization of the operation and interface of at least one application in response to a change in user preference.

BACKGROUND OF THE INVENTION

Generally speaking, software applications consist of many files, all of which interoperate to provide the overall functionality of, and the user interface to, the application. Applications can be executables, scripts, and/or libraries such as dynamic link libraries (DLLs), or any combination thereof, which may be provided with resources that support the application's functionality and interface features. At present, DLLs, which are libraries of executable code and/or data, are commonly used by programmers to provide one or more of the functions of an application. The application dynamically links to the DLLs, which are often distributed with the application. In some instances, DLLs may be shared among applications or made available by an operating system. Therefore, both the DLLs and the application code which is programmed to link to the DLLs must be present on the computer at runtime for the application to function properly.

As used herein, the term "resource" or "resources" is used in accordance with its ordinary meaning and generally refers to a library containing text strings and/or image data processed by an application, such as for example those needed to generate a portion of the user interface. The resources of an application may, for example, be encompassed in one or more DLLs, text files, executables, or data files of any type.

Increasingly, however, due to the emergence of global communication networks, such as the Internet and/or the World Wide Web (the "Web"), the global distribution of computer applications has increased markedly. Through electronic file transfer protocols, web-based application providers can distribute applications to users in many different countries. Undoubtedly, these applications will require updates and patches, which may require additional resources. Moreover, it is likely that users will have different preferences related to the operation of the application.

Currently, applications, such as executables, Active X Controls, and the like, are downloaded to a client computer where the application is processed. In many instances, however, the application accesses the Internet or other communication network to retrieve information or content, such as stock quotes, news, electronic mail, and the like. Such "web-based" applications have become popular for a variety of functions because they have the ability to extract and display information in real-time or near real-time from remote databases.

Users desiring such web-based applications commonly register with an on-line application/content provider and download the application to the user's computer, typically referred to as the client computer. During the registration process, the user may be asked to enter personal information or preferences so that the operation and interface of the application can be customized. For instance, users from different countries may have a preference for which language (i.e., English, German, French, etc.) the application utilizes to generate the graphical user interface (GUI) and to communicate with the user. Indeed, in many instances, a user may download a series of applications which may each have certain similar customizable aspects. Furthermore, these applications may utilize the same resources and may be amenable to similar updates.

The presently known methods and systems for managing such resources and for permitting the customization of the operation and interface of applications have

shortcomings. For example, known programming methods cannot effectively update or patch existing resources or integrate new resources not initially incorporated into an application without having to restart the application. Therefore, these methods and systems fail to satisfactorily permit the customization of an application to different user preferences. In addition, known methods and systems do not efficiently permit an application/content provider to distribute a single international application.

One known method and system requires that the resources for customizing the application be built (or hard-coded) into the application itself and distributed with each copy. This method, however, undesirably increases the size of the application and is unable to process resources that were not incorporated into the application at the time of download and installation. Because these web-based applications are generally downloaded from remote servers using file transfer protocols (FTP), increased file size adversely affects the file transfer performance, and this increased time factor may be a deterrent to a user downloading it, in addition to increasing necessary network overhead.

Another known method requires the development of separate versions of the application with support for each particular preference. For instance, in the case of language preferences, a developer could provide a separate version for every region to which the application would be distributed. Maintaining numerous versions, however, is time consuming and can result in increased costs to the developer and users.

Yet another known solution would be to build client-side scripts or server-side software applications, which run within the user's Internet browser. In use, only the version of the scripted application that corresponds to the user's particular preference would be downloaded. A change of preference would prompt the download of a separate version of the scripted application. Such scripted applications, however, such as, for example, JavaScript™, are limited by the functionality and extensibility of the scripting language,

which often does not have the same speed of operation and access to system functions as full application languages. The availability of bandwidth to users can also adversely affect the performance of scripted or server-side applications, because the scripted application must be downloaded to the client computer each time a preference change is requested. Lastly, server-side applications have limited scalability due to the limits on physical server space and memory and processing power.

There is thus a desire and unmet need to provide a method and system for more efficiently managing the resources of one or more applications. A further desirable and unmet need is to permit the customization of application resources that operates at runtime and that can be efficiently maintained. Furthermore, it is a desirable and unmet need to provide a system and method for permitting updates, patches and further customizable options to be integrated into the application resources, while keeping the application streamlined and preserving its efficiency.

SUMMARY OF THE INVENTION

The present invention is directed to a system and method which overcomes shortcomings in the prior art. In particular, a system and method is provided for dynamically managing the resources of an application and for customizing the operation and interface of a web-based application in response to a change in user preference.

According to the present invention, a resource manager is programmed to identify resources by unique resource identifiers, and in response to a request for such resources, to retrieve those resources from a server if the resources are not stored locally on the client computer. Once retrieved, the resource manager passes the resources to the application. In one preferred embodiment, referred to generally as the "pull" embodiment, the application recognizes the need to process a resource and request that resource or group of resources from the resource manager, which in turn determines the availability of the resource and retrieves it

from the server system if it is locally unavailable. In another preferred embodiment, referred to generally as the "push" embodiment, the server system receives a preference change request and the resource manager, with the next communication with the server system, receives the preference parameters associated with the preference change. The resource manager then looks in its map of resource identifiers to determine whether the corresponding resources are locally stored. If one or more resources are not locally stored, the resource manager retrieves, from the server system, either the entire group of resources corresponding to the preference parameters or that particular missing resource. In the meantime, the resource manager passes the locally stored resources to the application. The resource manager then passes any resources retrieved from the server system to the application, as necessary.

As used herein, the term "application" generally refers to a software program which is substantially executed by a computer located on the client side of the client-server architecture. The term "web-based application" refers generally to applications that at least partially derive the data for operation from a remote server and/or are capable of receiving data from a remote server or other computing or storage device remote from the client, either at the server location or at one or more locations in the network to which the client is connected. Such applications, for the purposes of the present invention, may be coded in any programming language known in the art. In addition, the applications may be delivered to the client side in any manner known in the art, such as via file download, electronic mail, or any other file transfer protocols that use networks of interconnected devices to transfer data. Furthermore, such applications may be distributed on any art-recognized machine readable storage medium, such as a compact disc, floppy disc, tape, or the like.

The present invention utilizes a resource manager stored on a client computer to determine whether the resources necessary to process a particular aspect of an application are

present on the client computer at the time of execution of the application. In use, when the subject application is first downloaded to the client computer, the resource manager is also downloaded and stored on the client computer. The resource manager may be an integral part of the application or may stand alone as its own library, such as an "in-process" component object model ("COM") server (DLL) or an "out of process" COM server (.exe). In each case, however, the downloaded application is programmed to interoperate with the resource manager to determine whether the necessary resources for its operation have also been downloaded or otherwise made present on the client computer.

In general, according to the present invention, the resource manager contains a library of ID maps associating a resource to a unique resource identifier. The resource identifier may be, for example, an alphanumeric code containing a resource group ID, resource ID, date ID, version ID, and the like, or any combination thereof.

In one preferred embodiment, a server system receives a request to change a preference from a client computer through a network. Based upon the request, the server system determines the resources necessary to accommodate the request. Next, the server system communicates the preference parameters to the resource manager through the network. Alternatively, the resource manager with the next communication with the server system pulls the preference parameters associated with the preference change from the server system. The resource manager uses the preference parameters to determine whether the resources associated with the group identifier are stored on the client computer. If each resource of the group of resources are found on the client computer, the resource manager loads the resources into the processing memory of the client computer and passes the resources to the application. If one or more resource is not found on the client system, the resource manager retrieves either the entire group of resources or that particular resource from the server system using the resource's unique identifier.

Alternatively, the application can be programmed to recognize the need for a resource or group of resources and notify the resource manager to retrieve the resources. In this embodiment, the application can identify the unique group IDs associated with a particular language, region, or preference (e.g., language, region, etc.), or the unique resource ID, if only one of the group is needed.

An advantage of the present invention is that as new versions of the resources are created or new regions are supported, the application provider can distribute these additional resources to the resource manager as needed. For example, an application provider providing additional resources for a web-based application may make the additional resources available for download to the client computer. Once received by the resource manager, the resources are preferably stored either in the memory space of the manager or the memory space of the application on the client computer. However, the resources can be stored on the server system and loaded temporarily into the memory of the client computer as needed.

The present invention provides advantages to both the application user and the application provider. For the application provider, new resources or versions of the resources can be incorporated into an existing application without the need to release an entirely new version of the application. In addition, in the case of web-based applications, the manager can be alerted by a server system that new resources are available, thereby causing the new resources to be downloaded to the client computer without interrupting the user's experience. Because the application is either notified of additional resources or knows to ask the manager when additional resources are needed, the application can be customized seamlessly at runtime without re-execution of the application.

Since the resources for an application are essentially libraries of data, the resources are often associated with a customizable feature of the application. The manager advantageously permits an application provider to store a set of resources pertaining to particular

customizable options on a remote server. Using the manager, the application provider can permit a user to select certain customizable preference settings.

In operation, a user may request to change a particular preference setting, such as, for example, the language used to generate the user interface. In response to the request, the application will determine whether or not the resource that is needed to generate the user interface in the new language has been downloaded to the client computer, as described above. If it has not, depending upon the embodiment utilized, the application will either notify the resource manager that additional resources are needed (i.e., call an exposed function of the resource manager) or await notification from the resource manager. The resource manager in turn, in a web-enabled application example, retrieves the necessary resources from a remote server.

In addition, the manager permits additional users of the same client computer to customize the application and the user interface to their particular preferences as will be further illustrated in the detailed description in connection with the drawing figures. Furthermore, because the application is streamlined, due to the lack of a need to hard-code all of the resources necessary to provide customization capabilities, the user saves download time and disk space and avoids the need to re-start the application.

In another embodiment of the present invention, the resource manager manages the resources for a plurality of applications stored on the client computer. In accordance with this embodiment, a method of managing the resources for a plurality of applications generally comprises mapping an identifier to a resource, receiving into a server system a request to customize an application, communicating the identifier to a resource manager stored on a client computer, determining whether the resource associated to the identifier is stored on the client computer, and retrieving the resource if the resource is not stored on the client

computer. Next, each application that shares the resource is notified of the resource's availability, i.e., its accessibility to be linked to by a particular application.

In addition, a client-server system for managing the resources of one or more applications generally comprises a server system having a storage device, the storage device having stored thereon a plurality of resources for use with one or more applications, and a client computer having stored thereon the one or more applications and a manager. The client computer is inter-operative with the manager so as to register the application modules with the manager when the application module is downloaded to the client computer.

Other objects and features of the present invention will become apparent from the following detailed description, considered in conjunction with the accompanying drawing figures. It is to be understood, however, that the drawings are designed solely for the purpose of illustration and not as a definition of the limits of the invention, for which reference should be made to the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawing figures, which are merely illustrative, and wherein like reference characters denote similar elements throughout the several views:

FIG. 1 is a schematic overview of the system architecture of a preferred embodiment of the present invention;

FIG. 2 is a schematic block diagram of a client computer connected to a network and upon which the present invention may be implemented;

FIG. 3 is a flow diagram of a method of managing the resources of an application in accordance with a preferred embodiment of the present invention;

FIG. 4 is a flow diagram of a method of seamlessly integrating an application update in accordance with a preferred embodiment of the present invention;

FIG. 5 is a flow diagram of a method of customizing the operation and interface of an application in accordance with a preferred embodiment of the present invention; and

FIGS 6-10 are exemplary screenshots of a method of internationalizing an application in accordance with the various preferred embodiments of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

According to the present invention a system and method for managing the resources for one or more applications is provided. In operation, the present invention permits the resources of one or more applications to be updated, supplemented, or patched. Furthermore, the present invention advantageously permits additional resources to be downloaded and accessed by one or more applications, thereby permitting the internationalization of a suite of applications.

With reference generally to FIGS. 1-10, a resource manager 75 for use with the present invention is provided along with, prior to, or after delivery of the operating code of an application 70. Preferably, when the application 70 is loaded or downloaded to a client computer 50, the resource manager 75 is also loaded onto the client computer 50. In the alternative, the resource manager 75 can be hardcoded into the application 70 as an integral part. Thus, as will be described further, the manager 75 can be a standalone program or can be integrated within the application 70. As previously defined, the term "resource" generally refers to a library containing the text strings and/or image data processed an application 70, for example, to generate the user interface of an application. In use, the application links to the resource 72 either statically or dynamically (as needed), to process the application 70.

Referring now to FIG. 1, an overview of a preferred embodiment of the system architecture for use with the present invention is shown. A server system 10 of an application/content provider is connected to a global communications network 100 via a

generally known network interface. A plurality of client computers 50 are interconnected to the server system 10 through the network 100.

The server system 10 includes one or more storage devices 15, 20 for storing the content and data that is distributed through the network to the client computers 50. The server system 10 also includes a processor, such as the Intel Pentium III Xeon processor, memory (RAM, ROM), clocks, and other components commonly included with server systems known in the art (not shown). The storage device 15, 20 is any machine-readable storage medium now known or that will be known in the art, such as by way of non-limiting example, a hard disk, optical drive, or tape drive. The particular architecture of server system 10 is not critical to the invention and, therefore, server system 10 is not depicted in detail. The network 100 is preferably the Internet; although one skilled in the art will recognize that the network may be any communications network now known or that will be known that is capable of transmitting data, such as, for example, an intranet, local area network, wide area network, or other network using point-to-point protocols (PPP), wireless application protocols (WAP), and the like.

With further reference to FIG. 2, a schematic of a client computer 50 for use with the present invention is shown. The client computers 50 are any type of personal or network computer having an operating system and, preferably, running a browser such as Microsoft Internet Explorer or Netscape Navigator. Client computers 50 generally access the network 100 to download an application 70 from the server system 10. The application 70 is then stored and processed on the client computer 50.

Client computer 50 includes an internal bus 64 that facilitates communication of data between and among the various components of the client computer 50 and that also facilitates communication between the client computer 50 and external devices and systems via the network interface 68. A processor 66 is coupled to the bus 64 to process data, including

application programs, within the client computer 50. The client computer 50 also includes a main memory 60, such as, for example RAM or other equivalent dynamic memory storage, coupled to bus 64 for receiving and storing instructions communicated by the processor 66. Read-only memory (ROM) is also provided and coupled to the bus 64 to store static data and instructions for use by the processor.

Various input and output devices 54, 56, 58 are provided with the client computer 50, including, by way of non-limiting example, a display 54 (e.g., cathode ray tube (CRT), liquid crystal display (LCD), etc.), and an input device 56 (e.g., a keyboard, mouse, touch pad, or light pen). A second storage device 58 such as, for example, a magnetic disk drive and magnetic disk, a CD-ROM drive and CD-ROM, DVD, or other equivalent device, is coupled to the bus 68 for communication with the processor 66, main memory 60 and network interface 68. The specific hardware combination/configuration is not crucial to the instant invention, and may vary as a matter of design choice within the functional parameters taught herein.

Referring again to FIG. 1, a user (not shown) generally utilizes client computer 50 to access the network 100. Although, as set forth above, network 100 is not limited to the Internet, the present invention will be described in connection with use of the World Wide Web or Internet. The client computer 50 connects to the network 100 via the network interface 65 over a transmission media such as, by way of non-limiting example, coaxial cable, copper wires, or fiber optical cables. Communication between the client computer 50 and the network 100 may also be via wireless, satellite, or cellular interface. The network interface 68 facilitates two-way communication between the client computer 50 and the server system 10 provided by the content/application provider (not shown).

Content and information is downloaded to the client computer 50 through network 100 using, for example, the hypertext transfer protocol (HTTP) in various known formats,

such as, for example, HTML, DHTML, XML, scripting languages, and the like. A browser application is stored on the client computer 50 and is programmed to interpret the code downloaded from the server system 10, thereby producing a GUI with which the user can interact. By way of non-limiting example, applications 70 and related resources 72 are preferably downloaded using known hyper-text transfer protocols (HTTP). Once downloaded, the application 70 may be launched by the user or it may be self-extracting and self-executing. Resources 72 are stored on the client computer 50 as described below so as to be available to the application 70 when needed.

With reference now to FIG. 3, a method of managing the resources 72 of an application 70 is there depicted in accordance with an embodiment of the present invention, and designated generally as 300. At step 302, after the user has caused the client computer 50 to link to the server system 10 of the application provider through the network 100, the user downloads the application 70 from the provider's server system 10. The application 70 is downloaded along with a resource manager 75, which is either provided as an integral part of the application 70 or as a stand-alone component. Once the download is completed, the application 70 and resource manager 75 can be installed, in a step 304, either by manually launching an installation program or by automatically launching a self-extracting file. The resource manager 75 is preferably stored locally on the client computer 50 in a memory space reserved for the operation of the resource manager 75. Resource manager 75 is preferably a dynamic library, such as an in-process COM server (e.g., a DLL) or an out-of-process COM server (e.g., a EXE). As previously discussed, the resource manager 75 can also be integrated within the application 70. In either case, resource manager 75 is programmed to identify, map and store a unique identifier for each resource 72 that is downloaded to the client computer 50 and related to application 70. In short, the resource manager 75 maintains a map of the associations of unique identifiers to resources 72.

As described above, in a preferred embodiment, at step 306, the resource manager 75 maintains a map of unique identifiers for each resource 72 that is downloaded with the application 70. In this way, the resource manager 75 can identify all of the resources 72 available to the application 70 at any given time. Upon being prompted, as will be further described, in a step 308, the resource manager 75, at step 310, retrieves the resources 72 and send a notification message to application 70, which includes a pointer to the resources 72, so as to enable application 70 to link to resources 72 as needed.

The unique identifier can be formatted as a unique series of characters, numbers or other data readable by the manager 75, such as, for example a software identifier and unique number combination. By way of non-limiting example, the unique identifiers may include varying levels of identifiers such as regional identifiers, group identifiers, and individual resource identifiers. Furthermore, the unique identifiers can include date and/or version identifiers such that the resource manager 75 can identify updated versions of resources 72. Alternatively, a single unique identifier can be assigned to each resource 72, which includes the codes necessary to identify the resource 72 with a particular region, language or group.

As used herein, the term "request" refers to a message or data transmitted between the client computer 50 through network 100 to server system 10, so as to enable the communication of resource identifiers and other types of communication.

The process of dynamically linking to the resources will now be described. In one preferred embodiment, upon retrieving the resources 72 as previously described, resources manager 75 calls a function exposed by application 70 and passes the resources 72 to application 70 as an argument. One skilled in the art will recognize that the linking procedure may be performed in many ways, including without limitation, the use of pointers, addresses, and the like.

As such, the application 70 is programmed to receive the notification from the resource manager 75 that additional resources 72 are available. If the conditions are such that the additional resource 72 is needed (i.e., the resource is needed to generate a user interface associated with a particular preference setting), the application 70 will link, as describe above, to the resource 72 so that the text-strings and image data contained within the resource 72 can be accessed and used by the application 70. Since the download and notification procedure is generally relatively short, the resource 72 will be available for access substantially immediately after it is downloaded. In this way, the present invention permits the application 70 to be updated or customized substantially in real time. Furthermore, because the application 70 is programmed to receive the notification from the resource manager 75 (or check with the manager for new resources 72 when the application 70 is prompted for that resource 72), there is no need to re-execute the application 70 upon the availability of new resources 72. The resources 72 therefore can become available to the application 70 during its execution.

Referring now to FIG. 4, in yet another preferred embodiment, the application 70 calls a function exposed by the resource manager 75 to retrieve the resources 72, upon the retrieval of which, the resource manager 75 returns the resources 72 to the application 70. The application 70 is programmed to identify a need for resources 72 and check with the resource manager 75 each time it is necessary to link to a resource 72, as in step 402. In step 404, the application 70 calls a function exposed by the resource manager 75. If the resource manager 75 determines that the resource is stored on the client computer 50, in a step 406, the resource manager 75 passes the resource 72 to application 70, in a step 408A. If, however, the resource 72 has not been found, the resource 72 is retrieved or downloaded from the server system 10, in step 408B. In both instances, the resource manager 75 returns a pointer to application 70 to enable linking to resources 72.

Because querying the resource manager 75 each time a resource 72 is needed can slow the procedure, it is preferable to pass newly available resources 72 to the application 70 and to store a copy of the resource 72 in the memory space of the application 70. In this way, the application 70 can process instructions to link to available resources 72 without waiting for a response from the resource manager 75.

However, one skilled in the art will recognize that the resources 72 may be stored on a server system 10 and downloaded to client computer 50 only as needed. As such, resources 72 would only be temporarily loaded into the processing memory space of client computer 50 when needed to generate a user interface.

As will be shown and described with reference to the various embodiments described herein, the method of FIG. 3 may be adapted to many different applications in accordance with the present invention.

For instance, a preferred embodiment of the present invention may be adapted to a method of downloading an updated resource. By way of example, during operation, the resource manager 75 can be notified by the application provider of an updated resource 72, which requires additional resources 72. In the alternative, the user can request to download the updated resource 72 from the server system 10. The resource manager 75 determines which version of the resource 72 is stored locally on client computer 50 by comparing the unique identifiers of the resources 72, which may include a date or version ID. Once retrieved, the resource 72 will be linked to as described above.

Referring to FIG. 5, a method of customizing the operation or interface of an application 70 is there described in accordance with an embodiment of the present invention, and generally designated as 500. At step 502, a user, who has previously registered with the application provider, logs into the application provider's site using a password or other unique identifier that is recognizable by the server system 10. During the aforementioned previous

registration, the user may have been prompted by the application/content provider to input certain personal information and preference settings, such as by way of non-limiting example a language preference. This information is stored in a database on the server system 10. In step 504, the unique user identifier is used by the server system 10 to retrieve the preference settings or other such customizable information from the server database. In a step 506, an access code or a group ID associated with a group of resources 72 is communicated to the client computer 50 and identified by the resource manager 75.

At step 508, the resource manager 75 determines the availability of the resource 72 required to display the application interface in accordance with the user's registered preferences. If the resource 72 is available, in a step 510, the resource manager passes the resources to application 70 as described above, to enable application 70 to generate the customized functionality. If, however, the resource 72 is not yet available, the resources manager 75 retrieves the needed resource 72 from the server system 10, in a step 512.

Such employment of the present invention enables several registered users having different preferences to use the same application 70 without having to re-customize the application's operation and interface. For example, a first user may have registered with an English language preference. Each time that user launches application 70, the user interface is generated using the resources corresponding to the English language parameters, as shown for example in FIG. 6. If a second user, who had registered with a Spanish language preference logs onto the application provider's web site using the same client computer 50 as the first user, the user interface will dynamically load in the Spanish language, as shown in FIG. 9. In operation, as described above, the server system 10 retrieves the second user's preference settings so that the resource manager 75, in its next communication with the server system 10, can receive the new Spanish language parameters. Using the parameters, the resource manager 75 determines whether the resources 72 needed to generate the Spanish

language interface are locally stored. If the resources 72 are not locally stored, the resource manager 75 retrieves them from the server system 10. In this way, the present invention can dynamically load a user interface for multiple users of the same client computer 50. In addition, because the resource files are generally relatively small, they can be downloaded quickly to the client computer 50 and seamlessly incorporated into the application 70 at runtime.

With reference to FIGS. 3-5, it can be seen that although depicted and described with reference to one application 70, the methods of the present invention can be adapted to manage the resources of two or more applications 70. By way of example only, the customization of the user interface to a particular language preference may be applicable to several applications 70 stored on a particular client computer 50. As such, a group of applications 70 can be internationalized by using the resource manager 75 of the present invention to manage the resources 72 of several applications 70.

Therefore, with further reference to FIGS. 1, 2, and 6-10, there is shown and herein described an exemplary method of customizing an application 70 and, in particular a method of internationalizing an application 70. As shown in FIGS. 1 and 6, a web site of a content provider is delivered to the display device 54 of the client computer 50 through the network 100. A browser application 110, such as Microsoft Internet Explorer or Netscape Navigator, interprets the downloaded content, which is in the form of images, HTML code and the like, scripting languages (e.g., JavaScript, VBScript, etc.), and applets, to name a few, to generate a user interface in window 115. A component object model (COM) technology, such as for example an ActiveX Control or a Plug-in, controls and/or modifies a portion of the browser interface as shown in toolbar 130. Such a browser interface overlay is disclosed in Applicant's co-pending U.S. Application Serial No. 09/429,585, the entire disclosure of which is incorporated herein by reference. The COM application links to library resources, such as

DLLs, to generate the particular interface of the tools within the toolbar 130. For example, as shown, in FIGS. 6 and 7, the English language is used in the buttons 132, drop down menus 134, and message boxes 136.

As described above, however, it may be desirable for another user of the same application 70 or users in different countries to change the language setting to their preferred language. With further reference to FIG. 8, a user is provided a preference setting screen 120 by the application/content provider. In this example, the user is changing the language setting to Spanish 190. One skilled in the art will recognize, however, that the particular type of preference setting (e.g., language, interface color, button type, tools available, etc.) for which the present invention is utilized is a matter of design choice; and, accordingly, the present invention can be adapted for any number of customizable preference settings pertaining to one or more applications 70 programmed to operate with the resource manager 75 of the present invention.

Referring again to FIG. 8, when the user has selected a preference, the user clicks the "Finished" button 122. for example, to complete the setting change. With further reference to FIGS. 9 and 10, the Spanish language replaces the English language in the buttons 132, drop down menus 134, and message boxes 136. It will be noted that the particular way in which a user can change the preference setting is not critical to the invention and the description herein is merely illustrative of one such way.

With reference again to FIGS. 6-10, toolbar 130 is generated by the execution of application 70 which links to and accesses resources 72, which include, for example, an ActiveX control or Plug-in. The resource 72 contains the code necessary to generate user interface and functionality of the toolbar 130. In use, application 70 is inter-operative with resource 72 to generate a shell within the browser 110 within which the functionality of the ActiveX control or Plug-in can be added to toolbar 130. By way of non-limiting example, the

data processed to build the user interface of toolbar 130 is defined by one or more resources 72 which contain the ActiveX control or Plug-in necessary to generate the interface. For instance, the strings of text that can be seen in FIGS. 6-8, which are in English, are stored in one of the resources 72 and provided to application 70 as needed to generate the user interface of toolbar 130. Similarly, the strings of text found in message box 136 as depicted in FIG. 7 are stored in one of the resources 72 and provided to application 70 as need to generate a particular message box.

Through the application provider's web site, illustrated for example in browser window 115, a user can select a preference setting. The preference setting, as one skilled in the art will recognize, may include many different types of customizable options, such as by way of non-limiting example language. With reference to FIG. 8, the user chooses to change his/her language preference from English to Spanish. Once the user has completed changing and reviewing his/her selections, the server system 10 either communicates the group identifier associated with the group of resources needed to accommodate the request to the client computer (as described above in connection with drawing figures 6 through 10) or the resource manager 75 receives a preference parameter corresponding to the resources from the server system 10 (as described in connection with drawing figure 5).

Upon receipt of the group identifier, resource manager 75 uses the group identifier to determine whether the group of resources associated with the group identifier is stored locally on the client computer. If, in the alternative embodiment, the resource manager 75 receives a preference parameter related to the region or language preference, the resource manager looks up the corresponding resource identifier in the resource manager's 75 map of identifiers to resources to determine whether the resources associated with the group identifier are stored locally on the client computer. Thus, if the group of resources 72 needed to generate the Spanish language toolbar 130 is not found on the client computer 50, the resource manager 75

retrieves them from the server system and passes the needed group of resources 72 to the browser application 70 so as to enable the application to access the group of resources 72, as described above. In this way, application 70 can link to the group of resources, load the text strings and image data into the memory of the client computer 50, and generate the toolbar 130 in the Spanish language.

Because the resources 72 for changing the preference setting are stored on the server system 10, a streamlined user application 70 can be delivered to the client computer 50 without having the entire library of resources 72 for each particular preference hard-coded into the application 70. Instead, the resources 72 are advantageously downloaded and dynamically integrated into the application 70 at runtime. In addition, because the resource manager 75 can manage resources 72 for more than one application 70, each application 70 can be customized in response to a change in the user preference. As such, applications can be internationalized, customized, updated, or patched in a seamless fashion without negatively impacting the user's experience. As such, there is no need to restart the application 70 upon the application 70 being internationalized, customized, updated, and/or patched.

Due to the increased convenience and functionality offered to a user, the present invention offers a means by which a web site can increase the frequency with which a user accesses the web site. According to this preferred embodiment, a user profile database 20 is maintained on the server system 10, which serves the web site to the browser application. The user profile database includes at least one customizable option. As used herein, the term "customizable option" or "preferences," refers to those options which the user may choose from and which are stored in the user profile on the server system. The server system, as described above, permits the user to change the customizable option. In response to the change in the customizable option, the client computer generates a browser interface on the client computer in the manner previously described.

Thus, while there have been shown and described and pointed out fundamental novel features of the invention as applied to preferred embodiments thereof, it will be understood that various omissions and substitutions and changes in the form and details of the disclosed invention may be made by those skilled in the art without departing from the spirit of the invention. It is the intention, therefore, to be limited only as indicated by the scope of the claims appended hereto.

What is claimed is:

1. A method of managing resources for a software application deployed in a client-server system, comprising:

(a) receiving at the server system a request from a client computer to change the software application;

(b) receiving into the client computer, from the server system, a preference parameter associated with a group of resources needed to accommodate the request;

(c) determining whether the needed group of resources is stored locally on the client computer; and

(d) retrieving the needed group of resources from the server system if the needed group of resources is not stored locally on the client computer.

2. The method of claim 1, further comprising:

(e) passing the needed group of resources to the software application so as to enable the application to access the group of resources.

3. The method of claim 2, wherein step (e) comprises:

calling a function exposed by the software application; and

passing the needed group of resources to the software application as an argument.

4. The method of claim 2, further comprising:

(f) generating a user interface on the client computer.

5. The method of claim 4, wherein step (f) comprises:

linking to the group of resources;

loading text strings and image data into memory of the client computer; and

producing a user interface using the text strings and image data.

6. The method of claim 1, further comprising:

(e) storing the needed group of resources in a memory space of the client computer; and

(f) passing the needed group of resources to the software application so as to enable the application to access the group of resources.

7. The method of claim 6, further comprising:

(g) generating a user interface on the client computer.

8. A method of internationalizing a software application deployed in a client server environment, comprising:

(a) storing on a server system a plurality of geographic region-specific groups of resources;

(b) receiving at a server system a request from a client computer that the software application exhibit characteristics indicative of a desired geographic region;

(c) receiving from the server system into a resource manager on the client computer a preference parameter associated with a region-specific group of resources needed to accommodate the request;

(d) determining whether the needed region-specific group of resources is stored locally on the client computer; and

(e) retrieving the region-specific group of resources from the server system if the needed region-specific group of resources is not stored locally on the client computer.

9. The method of claim 8, further comprising:

(f) passing the needed region-specific group of resources to the software application so as to enable the application to access the needed region-specific group of resources.

10. The method of claim 9, wherein step (f) comprises:

calling a function exposed by the software application; and

passing the needed region-specific group of resources to the software application as an argument.

11. The method of claim 10, further comprising:

(g) generating a user interface on the client computer.

12. The method of claim 11, wherein step (g) comprises:

linking to the group of resources;

loading text strings and image data into memory of the client computer; and

producing a user interface using the text strings and image data.

13. The method of claim 8, further comprising:

(f) storing the needed region-specific group of resources in a memory space of the client computer; and

(g) passing the needed region-specific group of resources to the software application so as to enable the application to access the needed region-specific group of resources.

14. The method of claim 13, further comprising:

(h) generating a user interface on the client computer utilizing the resources of the needed group.

15. A system for customizing a user interface of a software application stored on a client computer; comprising:

a communication link to a server system; and

a processor of the client computer operative with a software implemented resource manager to:

receive from said server system a preference parameter associated with a needed group of resources;

determine whether the needed group of resources associated with the preference parameter is stored on the client computer; and

retrieve the needed group of resources associated with the preference parameter from the server system if the needed group of resources is not stored locally on the client computer so that the software application is enabled to generate a user interface of the software application using the needed group of resources.

16. The system of claim 15, wherein the client computer receives the preference parameter from the server system in response to a user generated request to change a preference.

17. The system of claim 16, wherein the preference is a regional setting.

18. The system of claim 16, wherein the preference is a language setting.

19. The system of claim 15, wherein the generation of the user interface further comprises calling a function exposed by the software application, passing the needed group of resources to the software application as an argument, so as to enable the software application to link to the needed group of resources, load text strings and image data into a memory of the client computer, and generate the customized user interface using the text strings and image data.

20. A client-server system for permitting the internationalization of one or more software applications, comprising:

a server system having stored thereon a plurality of groups of resources, each of the plurality of groups associated with a preference, and the server system maintaining a user profile database for storing the preference of a user; and

a client computer interconnected with the server system through a network, the client computer having an input device for permitting the user to transmit preference indicative data to the server system and a software implemented resource manager for

receiving from the server system a preference parameter associated with the group of resources corresponding to the preference parameter, such that the resource manager can determine whether the group of resources are stored locally on the client computer.

21. The client-server system of claim 20, wherein the group of resources is retrieved from the server system if the group of resources is not stored locally on the client computer.

22. A method of modifying a toolbar interface of a browser application, comprising:

(a) generating on a client computer the toolbar interface using a first group of resources;

(b) receiving into a server system a request to change the toolbar interface;

(c) communicating, from the server system, a unique identifier associated with a second group of resources needed to change the toolbar interface to the client computer;

(d) determining whether the second group of resources associated with the unique identifier is stored locally on the client computer; and

(e) retrieving the second group of resources associated with the unique identifier from the server system if the second group of resources is not stored locally on the client computer.

23. The method of claim 22, further comprising:

(f) passing the second group of resources to the browser application so as to enable the browser application to access the second group of resources.

24. The method of claim 23, wherein step (f) comprises:

calling a function exposed by the browser application; and

passing the group of resources to the browser application as an argument.

25. The method of claim 23, further comprising:

(g) accessing the second group of resources;

(h) loading text strings and image data of the second group of resources into a memory of the client computer; and

(i) generating the toolbar interface using the text strings and image data.

26. The method of claim 22, further comprising:

(f) storing the group of resources in a memory space of the client computer; and

(g) passing the group of resources to the software application so as to enable the application to access the group of resources.

27. A method of increasing the desirability of a user accessible web site using a browser application and a client computer, the method comprising:

(a) maintaining a user profile database on a server system for serving the web site to the browser application, the user profile database including at least one customizable option;

(b) permitting the user to change the customizable option; and

(c) generating a browser interface on the client computer in response to the change in the customizable option without the need to restart the browser application.

28. The method of claim 27, wherein step (c) comprises:

communicating from the server system to the client computer a unique identifier associated with a group of resources needed to accommodate the change in the customizable option;

determining whether the group of resources associated with the unique identifier is stored locally on the client computer;

retrieving the group of resources associated with the unique identifier from the server system if the group of resources is not stored locally on the client computer; and

passing the group of resources to the browser application so as to enable the browser application to access the group of resources.

29. The method of claim 28 wherein step of passing the group of resources to the browser application comprises:

calling a function exposed by the browser application; and

passing the group of resources to the browser application as an argument.

30. A method of modifying a user's interaction with a software application deployed in a client server environment without the need to re-start the application, comprising the steps of:

(a) identifying a user interface preference of the user;

(b) identifying a preference specific resource necessary to the application to meet the user's preference; and

(c) making the preference specific resource available to the application.

31. The method of claim 30, wherein step (a) further comprises:

receiving into a server system a request to change a preference setting; and

communicating to a client computer from the server system a preference parameter identifying the user interface preference.

32. The method of claim 31, wherein step (b) further comprises:

retrieving a resource identifier associated with the preference specific resource identified by the preference parameter from a map of resource identifiers.

33. The method of claim 32, wherein step (c) further comprises:

determining whether the preference specific resource is locally stored using the resource identifier;

retrieving the preference specific resource from the server system, if the preference specific resource is not locally stored; and

passing the preference specific resource to the software application as an argument.

34. The method of claim 30, wherein step (a) further comprises:

receiving into a server system a request to change a preference setting; and

receiving into a client computer a preference parameter identifying the user interface preference during a subsequent communication with the server system.

35. The method of claim 34, wherein step (b) further comprises:

retrieving a resource identifier associated with the preference specific resource identified by the preference parameter from a map of resource identifiers.

36. The method of claim 35, wherein step (c) further comprises:

determining whether the preference specific resource is locally stored using the resource identifier;

retrieving the preference specific resource from the server system, if the preference specific resource is not locally stored; and

passing the preference specific resource to the software application as an argument.

37. A method of adapting a user interface of a software application in a client server environment to a user's specific language requirements without the need to restart the application, comprising:

(a) passing an item of data indicative of a user's language requirement to a server system;

(b) identifying the resources necessary to generate the user interface in the user's required language; and

(c) making the necessary resources available to the program.

38. The method of claim 37, wherein step (a) further comprises:

inputting into a client computer a user identifier associated with the user's language requirement; and

communicating the user identifier to the server system.

39. The method of claim 38, wherein step (b) further comprises:

receiving into the client computer from the server system a preference parameter identifying the necessary resources; and

retrieving a resource identifier associated with the necessary resources identified by the preference parameter from a map of resource identifiers.

40. The method of claim 39, wherein step (c) further comprises:

determining whether the necessary resources are locally stored using the resource identifier;

retrieving the necessary resources from the server system, if the necessary resources are not locally stored; and

passing the necessary resources to the software application as an argument.

41. A method of dynamically generating a language-specific user interface of a software application deployed in a client server environment, comprising:

(a) storing on a server system a plurality of language-specific resources;

(b) receiving at a server system a request from a client computer that the software application generate the language-specific user interface;

(c) receiving from the server system into the software application on the client computer a preference parameter associated with a language-specific resource needed to accommodate the request;

(d) calling a function exposed by a resource manager instructing the resource manager to return the needed language-specific resource to the software application;

(e) determining whether the needed language-specific resource is stored locally on the client computer;

(f) retrieving the needed language-specific resource from the server system if the needed language-specific resource is not stored locally on the client computer;

(g) passing the needed language-specific resource to the software application so as to enable the application to access the needed language-specific resource; and

(h) generating the language-specific user interface utilizing the language-specific resource.

42. The method of claim 41, wherein step (b) comprises:

accessing a user profile database;

changing a language setting stored in the user profile database; and

setting the preference parameter based upon the changed language setting.

43. The method of claim 41, wherein step (b) comprises:

logging onto the server system using a user identifier, the user identifier being associated with a language setting stored in a user profile database; and

setting the preference parameter based upon the stored language setting.

44. The method of claim 41, wherein step (g) comprises:

passing a pointer to the language-specific resource to the software application;

linking to the language-specific resource;

loading text strings and image data into memory of the client computer; and

producing the language-specific user interface using the text strings and image data.

45. A method of dynamically generating a language-specific user interface of a software application deployed in a client server environment, comprising:

(a) storing on a server system a plurality of language-specific resources;

(b) receiving at a server system a request from a client computer that the software application generate the language-specific user interface;

(c) receiving from the server system into a resource manager on the client computer a preference parameter associated with a language-specific resource needed to accommodate the request;

(d) determining whether the needed language-specific resource is stored locally on the client computer;

(e) retrieving the needed language-specific resource from the server system if the needed language-specific resource is not stored locally on the client computer;

(f) passing the needed language-specific resource to the software application so as to enable the application to access the needed language-specific resource; and

(g) generating the language-specific user interface utilizing the language-specific resource.

46. The method of claim 45, wherein step (b) comprises:

accessing a user profile database;

changing a language setting stored in the user profile database; and

setting the preference parameter based upon the changed language setting.

47. The method of claim 45, wherein step (b) comprises:

logging onto the server system using a user identifier, the user identifier being associated with a language setting stored in a user profile database; and

setting the preference parameter based upon the stored language setting.

48. The method of claim 45, wherein step (f) comprises:

calling a function exposed by the software application; and

passing a pointer to the language-specific resource to the software application.

49. The method of claim 48, wherein step (g) comprises:

linking to the language-specific resource using the pointer;

loading text strings and image data into memory of the client computer; and

producing the language-specific user interface using the text strings and image data.

FIG. 1

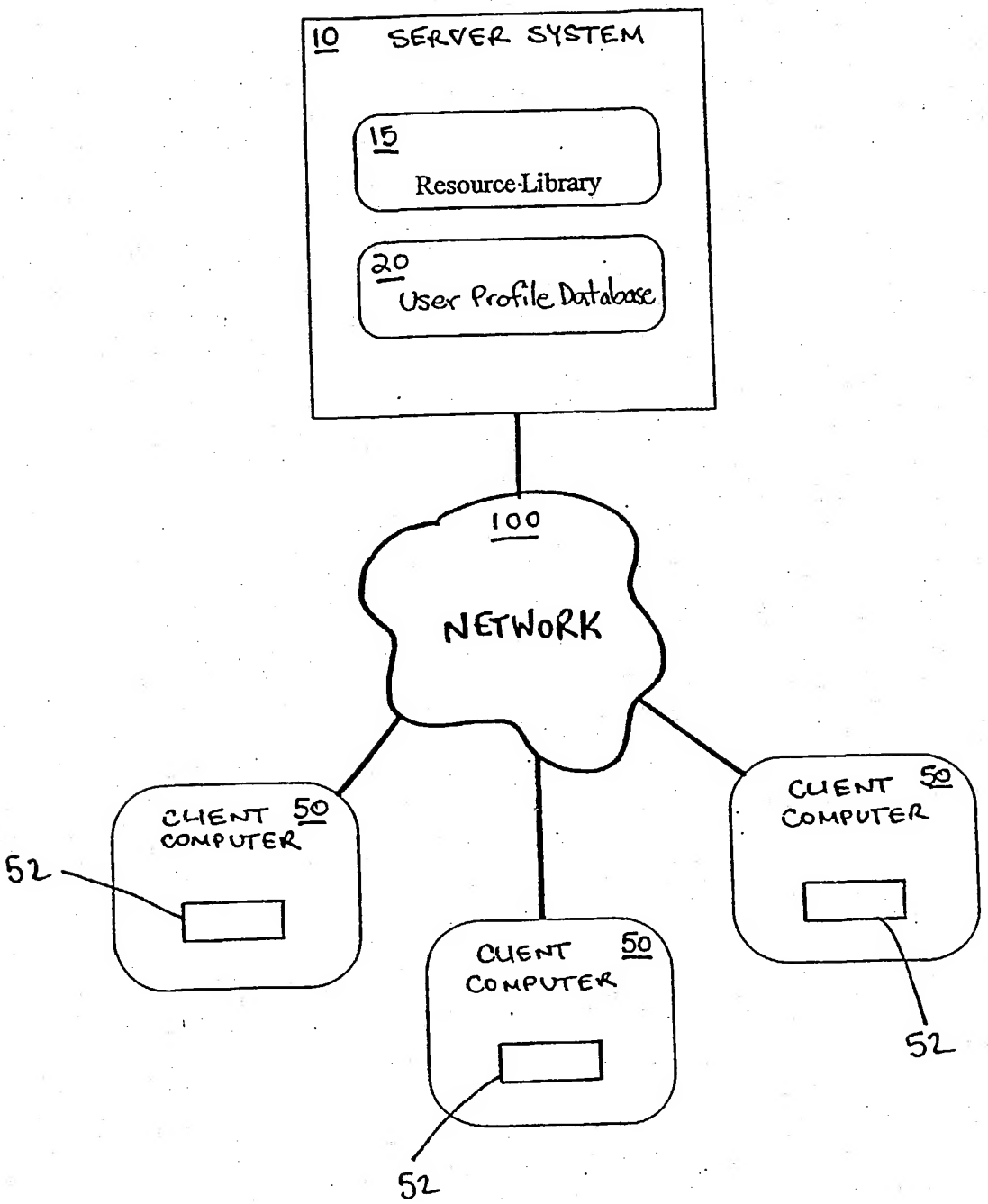

10 SERVER SYSTEM
15
Resource Library
20
User Profile Database
100
NETWORK
CLIENT 50
COMPUTER
52
CLIENT 50
COMPUTER
CLIENT 50
COMPUTER
52
52

FIG. 2

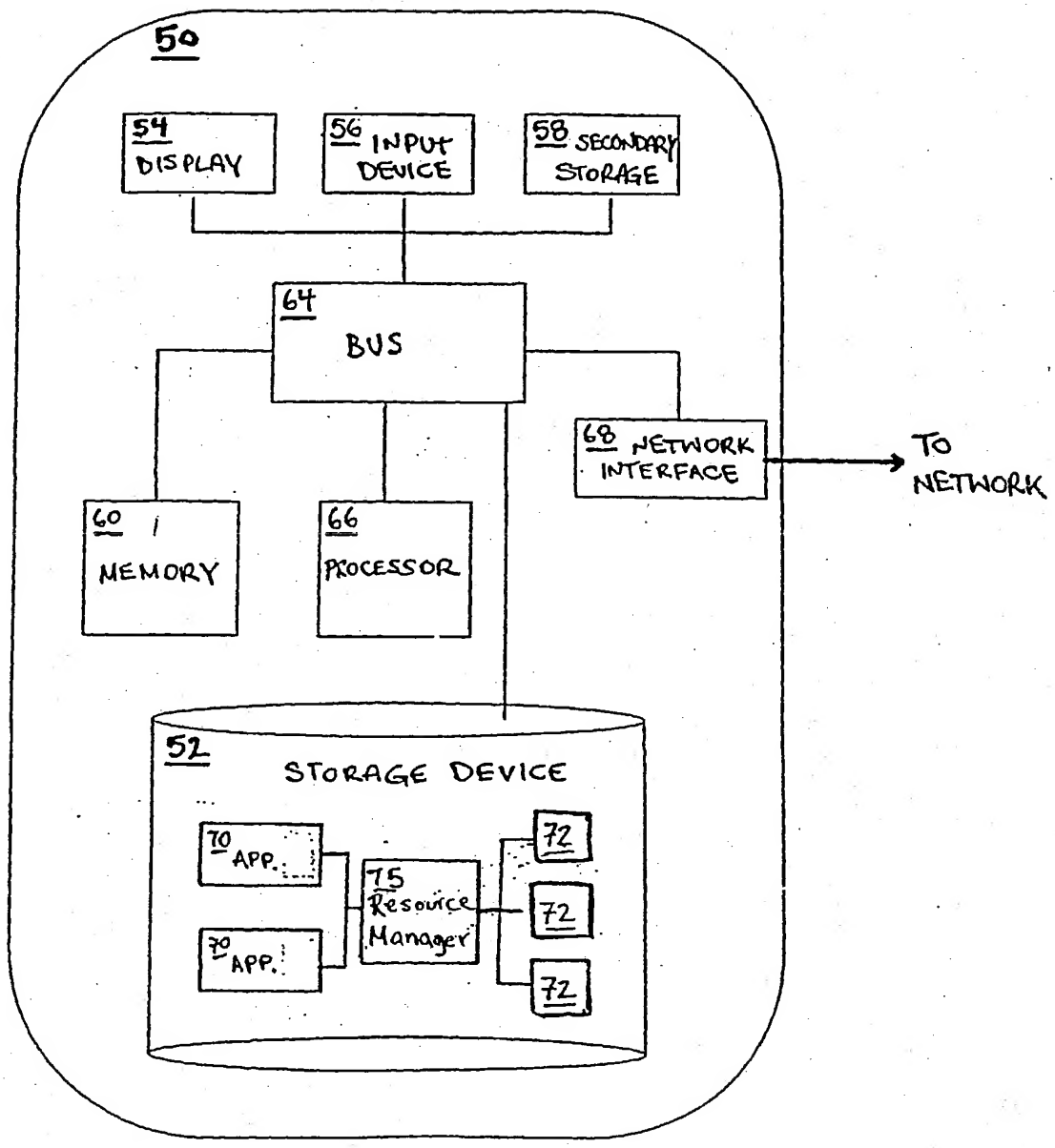

50
54 DISPLAY
56 INPUT DEVICE
58 SECONDARY STORAGE
64 BUS
68 NETWORK INTERFACE
TO NETWORK
60 MEMORY
66 PROCESSOR
52 STORAGE DEVICE
70 APP.
70 APP.
75 Resource Manager
72
72
72

FIG. 3

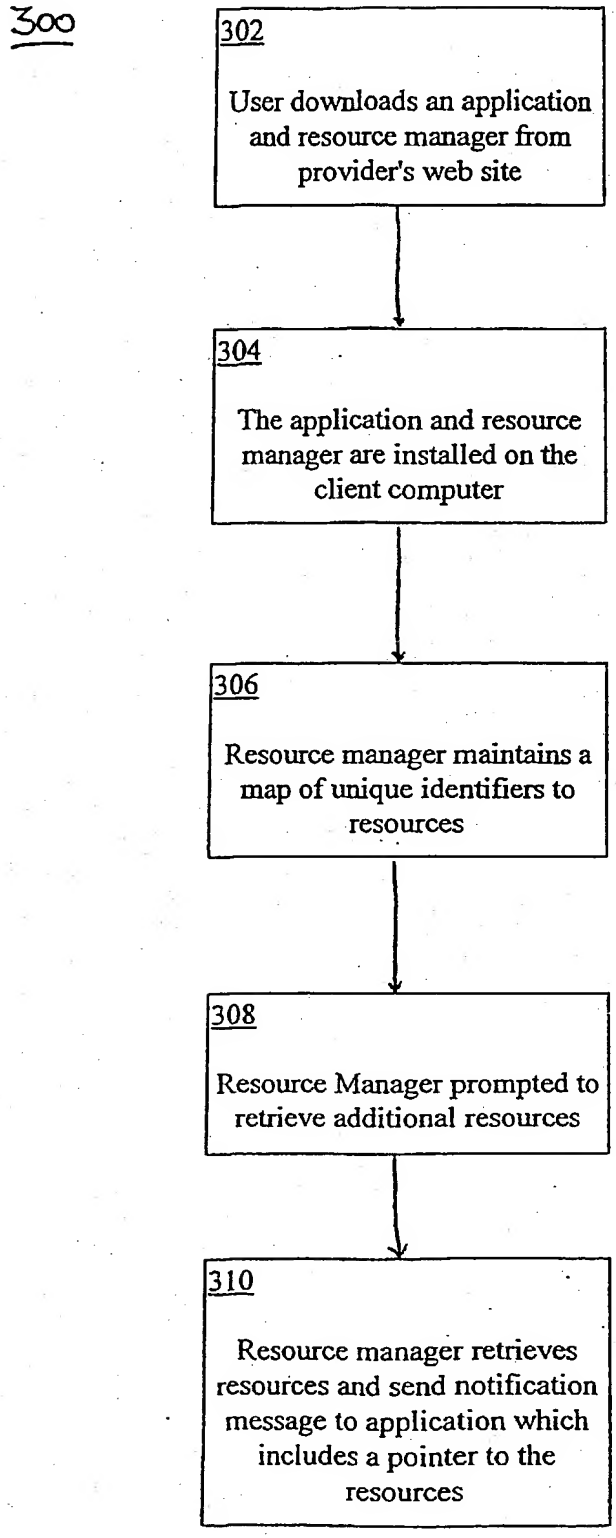

300
302
User downloads an application and resource manager from provider's web site
304
The application and resource manager are installed on the client computer
306
Resource manager maintains a map of unique identifiers to resources
308
Resource Manager prompted to retrieve additional resources
310
Resource manager retrieves resources and send notification message to application which includes a pointer to the resources

FIG. 4

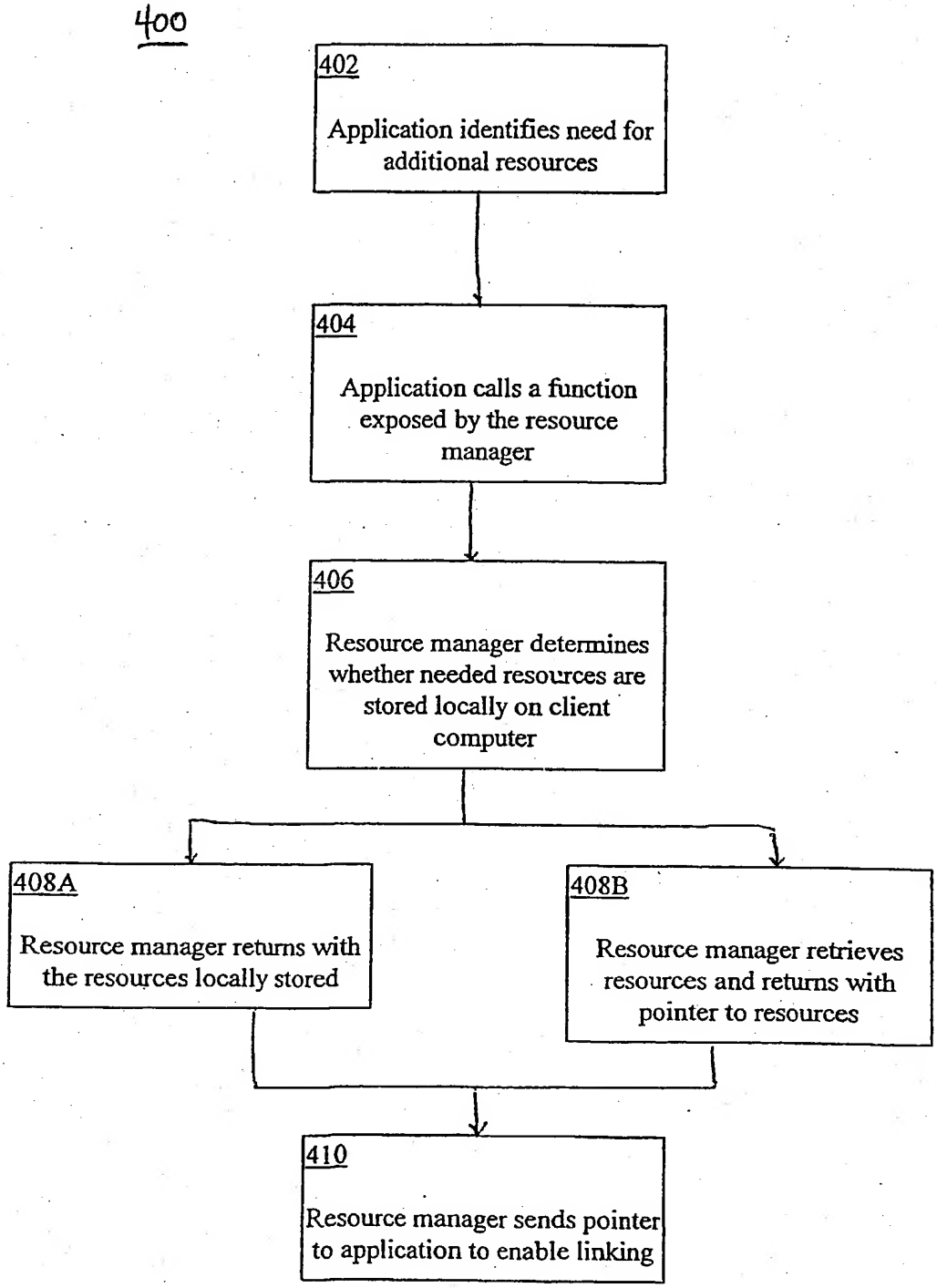

400
402
Application identifies need for additional resources
404
Application calls a function exposed by the resource manager
406
Resource manager determines whether needed resources are stored locally on client computer
408A
Resource manager returns with the resources locally stored
408B
Resource manager retrieves resources and returns with pointer to resources
410
Resource manager sends pointer to application to enable linking

FIG. 5

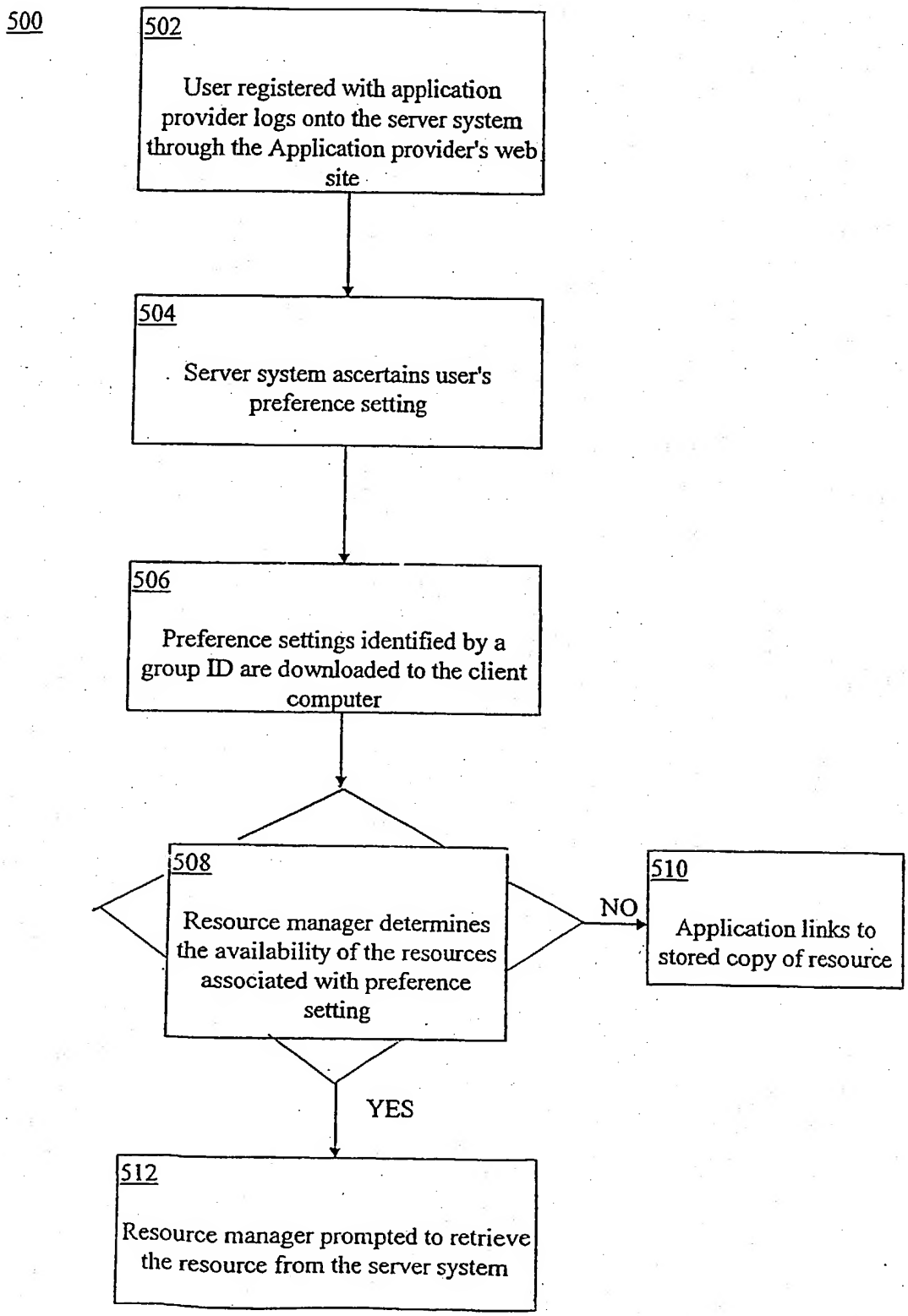

500
502
User registered with application provider logs onto the server system through the Application provider's web site
504
Server system ascertains user's preference setting
506
Preference settings identified by a group ID are downloaded to the client computer
508
Resource manager determines the availability of the resources associated with preference setting
NO
510
Application links to stored copy of resource
YES
512
Resource manager prompted to retrieve the resource from the server system

FIG. 6

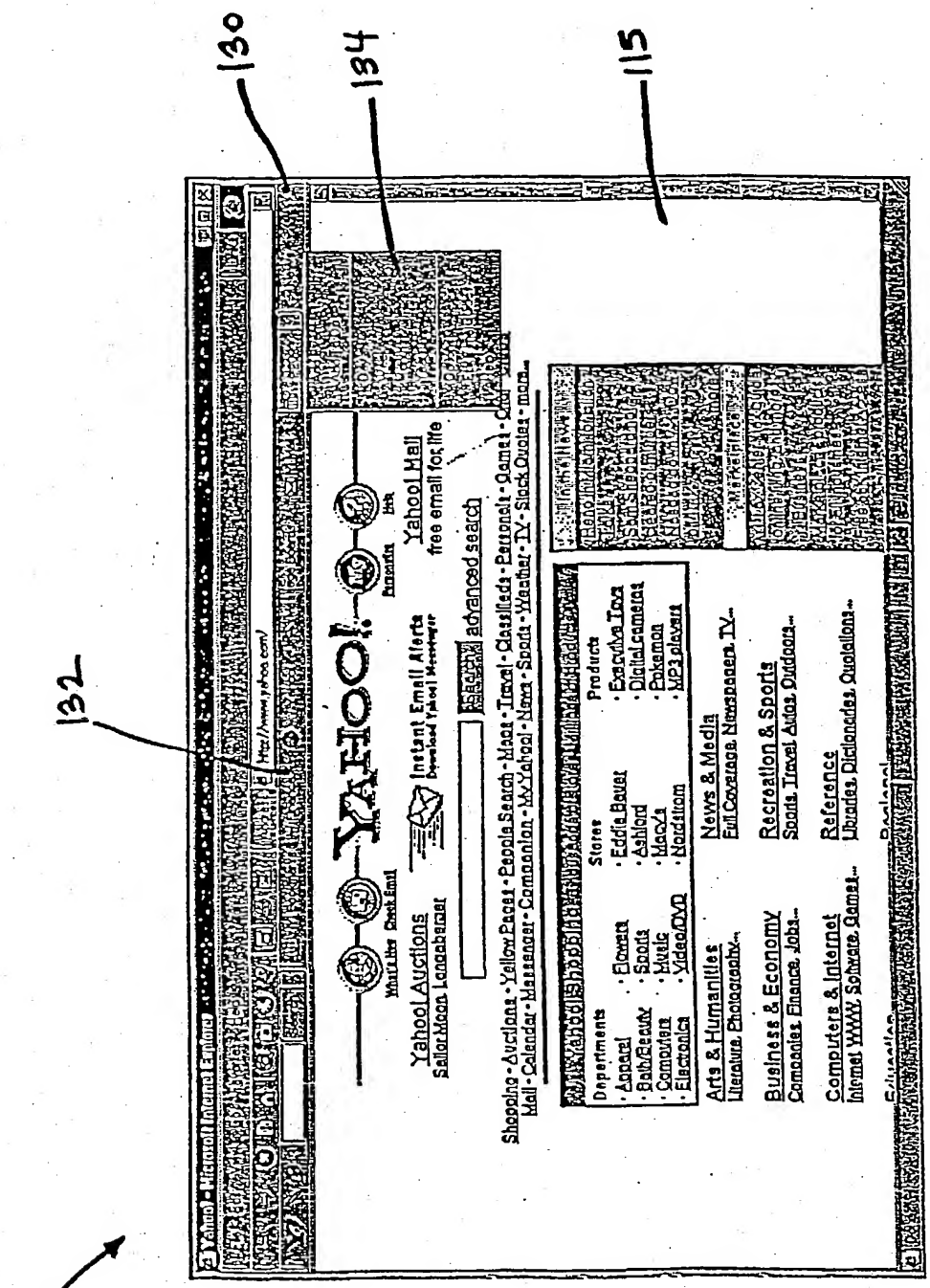

130
134
115
132
110
YAHOO!
Yahoo! Mail
Instant Email Alerts
Yahoo! Auctions
Sailor Moon Lunasbeamer
advanced search
Shopping - Auctions - Yellow Pages - People Search - Maps - Travel - Classifieds - Personals - Games - Chat
Mail - Calendar - Messenger - Companion - My Yahoo! - News - Sports - Weather - TV - Stock Quotes - more...
Departments
Apparel
Bath & Beauty
Computers
Electronics
Flowers
Sports
Music
Video/DVD
Stores
Eddie Bauer
Ashford
Macy's
Nordstrom
Products
Executive Toys
Digital Cameras
Pokemon
MP3 players
Arts & Humanities
Literature, Photography...
News & Media
Full Coverage, Newspapers, TV...
Business & Economy
Companies, Finance, Jobs...
Recreation & Sports
Sports, Travel, Autos, Outdoors...
Computers & Internet
Internet, WWW, Software, Games...
Reference
Libraries, Dictionaries, Quotations...

FIG. 7

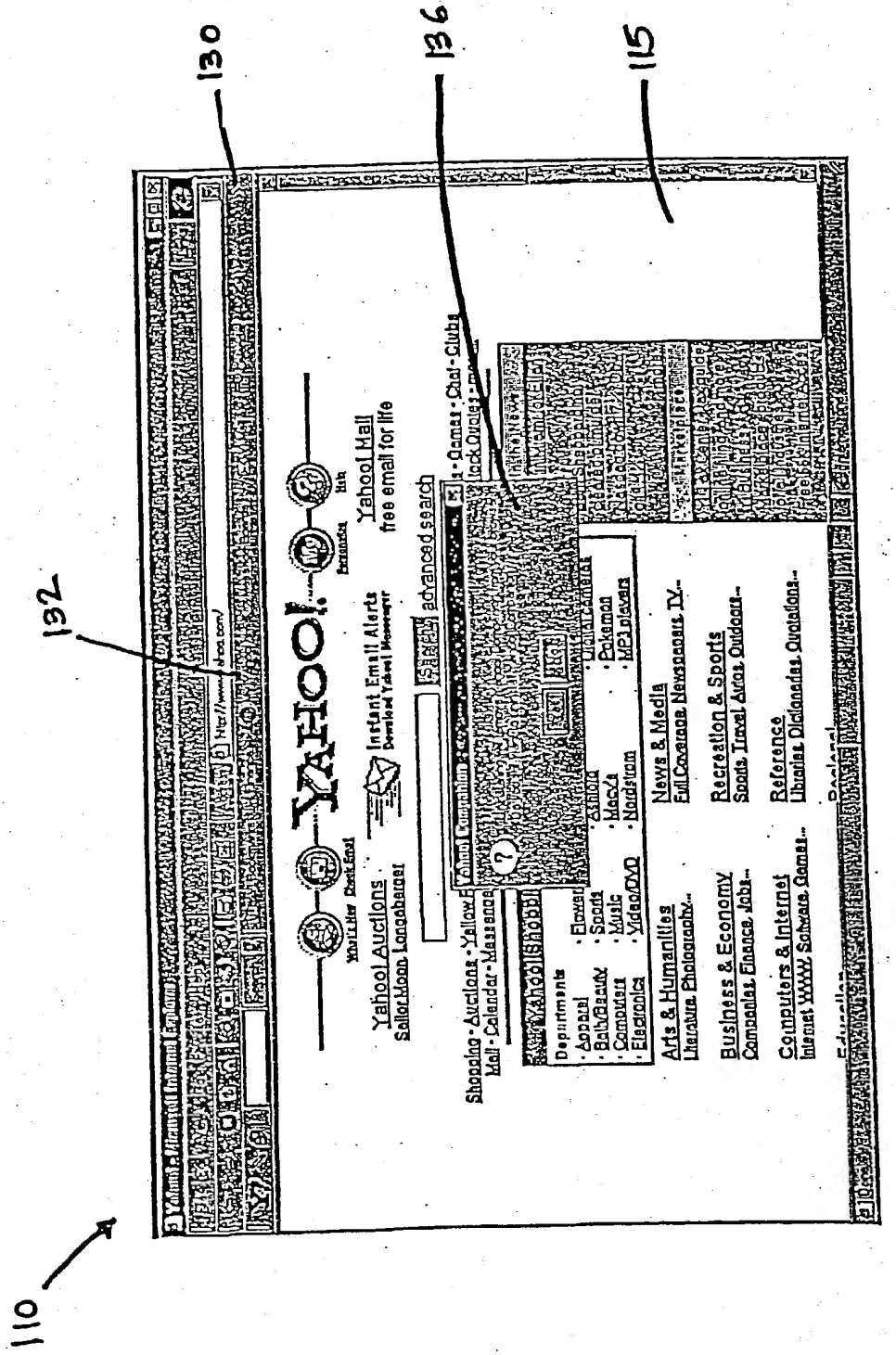

110
132
130
136
115
YAHOO!
Yahoo Auctions
Instant Email Alerts
Yahoo Mail
free email for life
advanced search
Shopping - Auctions - Yellow
Mail - Calendar - Messenger
Arts & Humanities
Literature, Photography...
News & Media
Full Coverage, Newspapers, TV...
Business & Economy
Companies, Finance, Jobs...
Recreation & Sports
Sports, Travel, Autos, Outdoors...
Computers & Internet
Internet, WWW, Software, Games...
Reference
Libraries, Dictionaries, Quotations...

FIG. 8

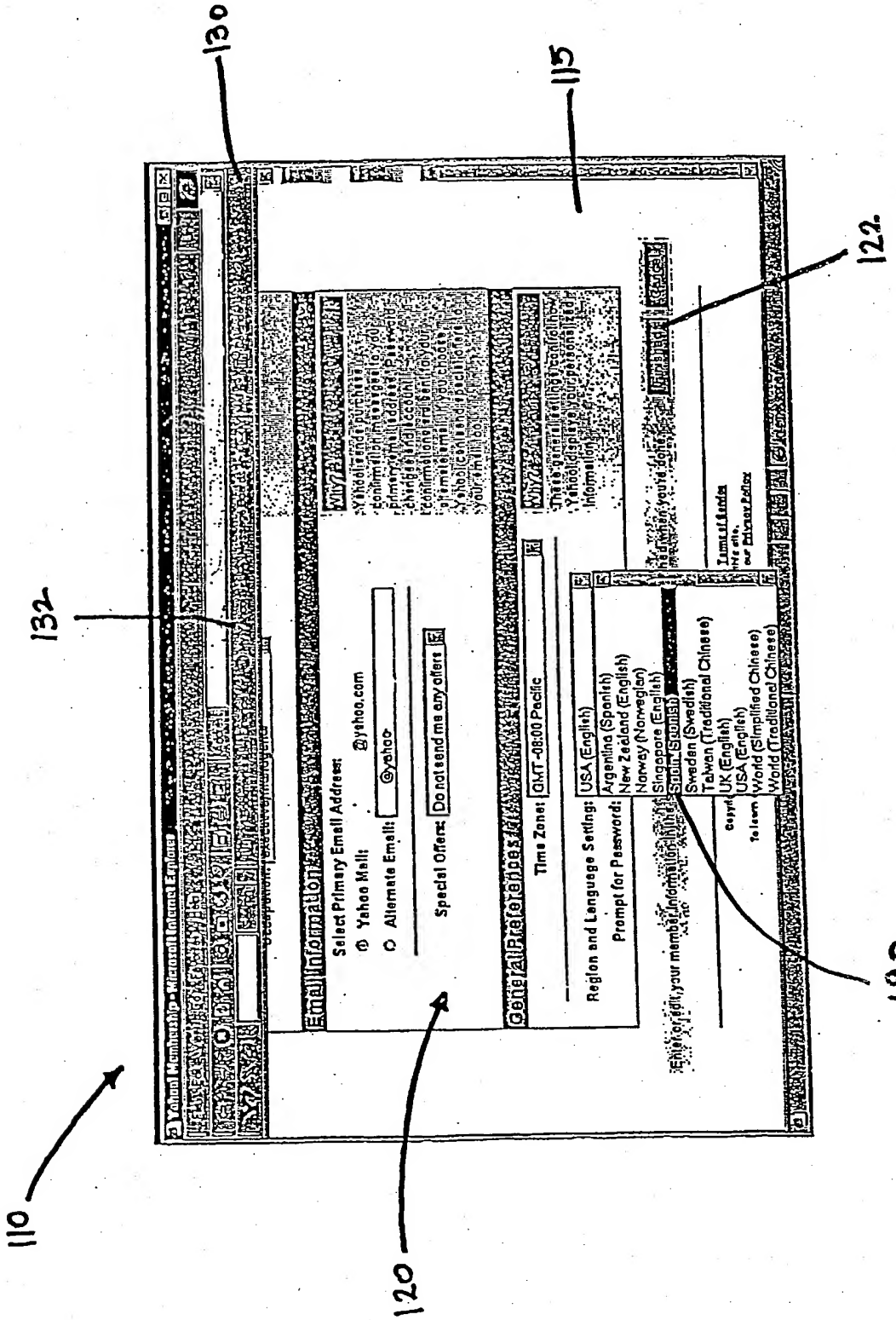

130
115
122
132
110
120
190
Email Information
Select Primary Email Address:
Yahoo Mail: @yahoo.com
Alternate Email: @yahoo
Special Offers: Do not send me any offers
General Preferences
Time Zone: GMT-08:00 Pacific
Region and Language Settings: USA (English)
Prompt for Password:
Argentina (Spanish)
New Zealand (English)
Norway (Norwegian)
Singapore (English)
South Korean
Sweden (Swedish)
Taiwan (Traditional Chinese)
UK (English)
USA (English)
World (Simplified Chinese)
World (Traditional Chinese)

FIG. 9

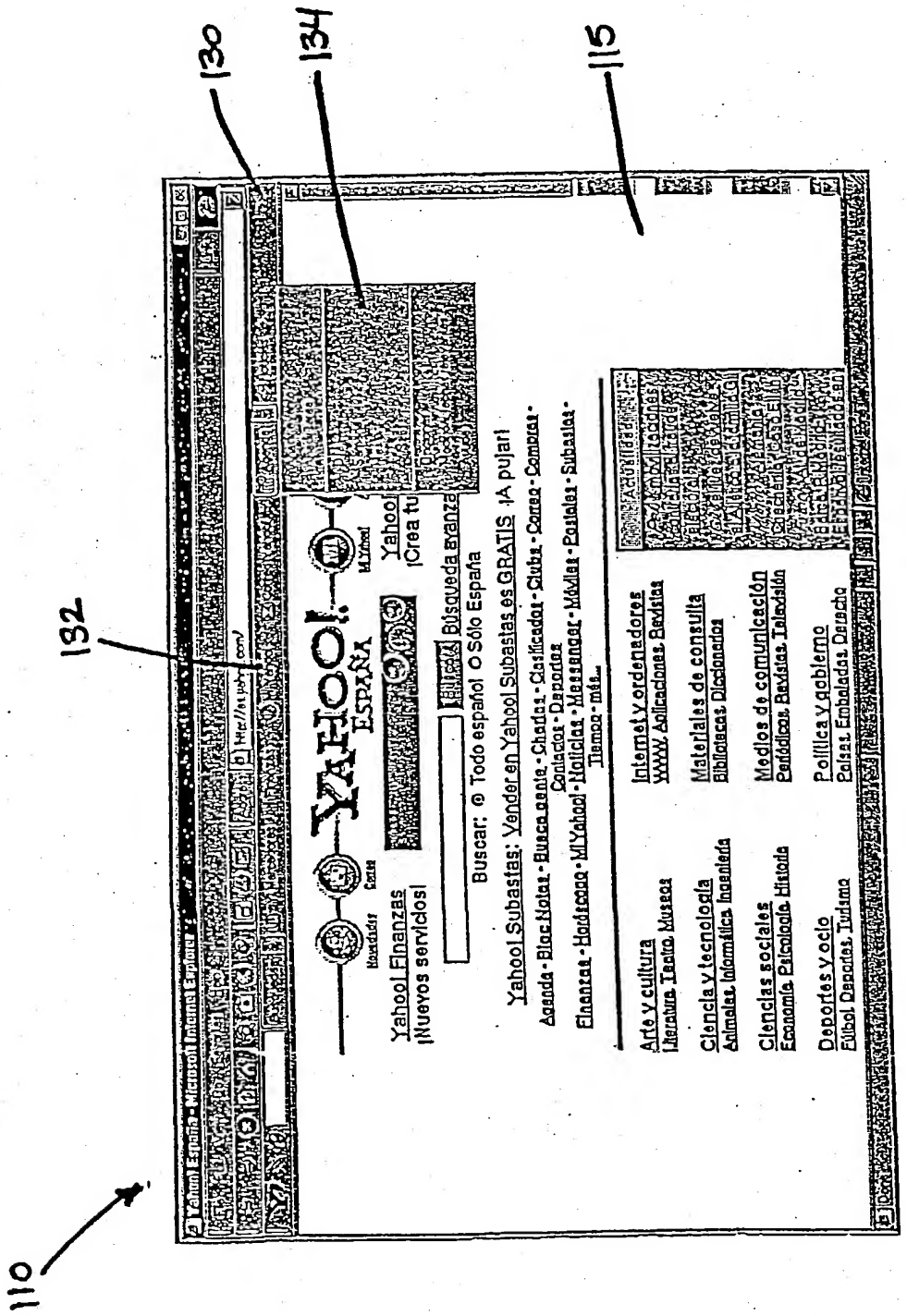
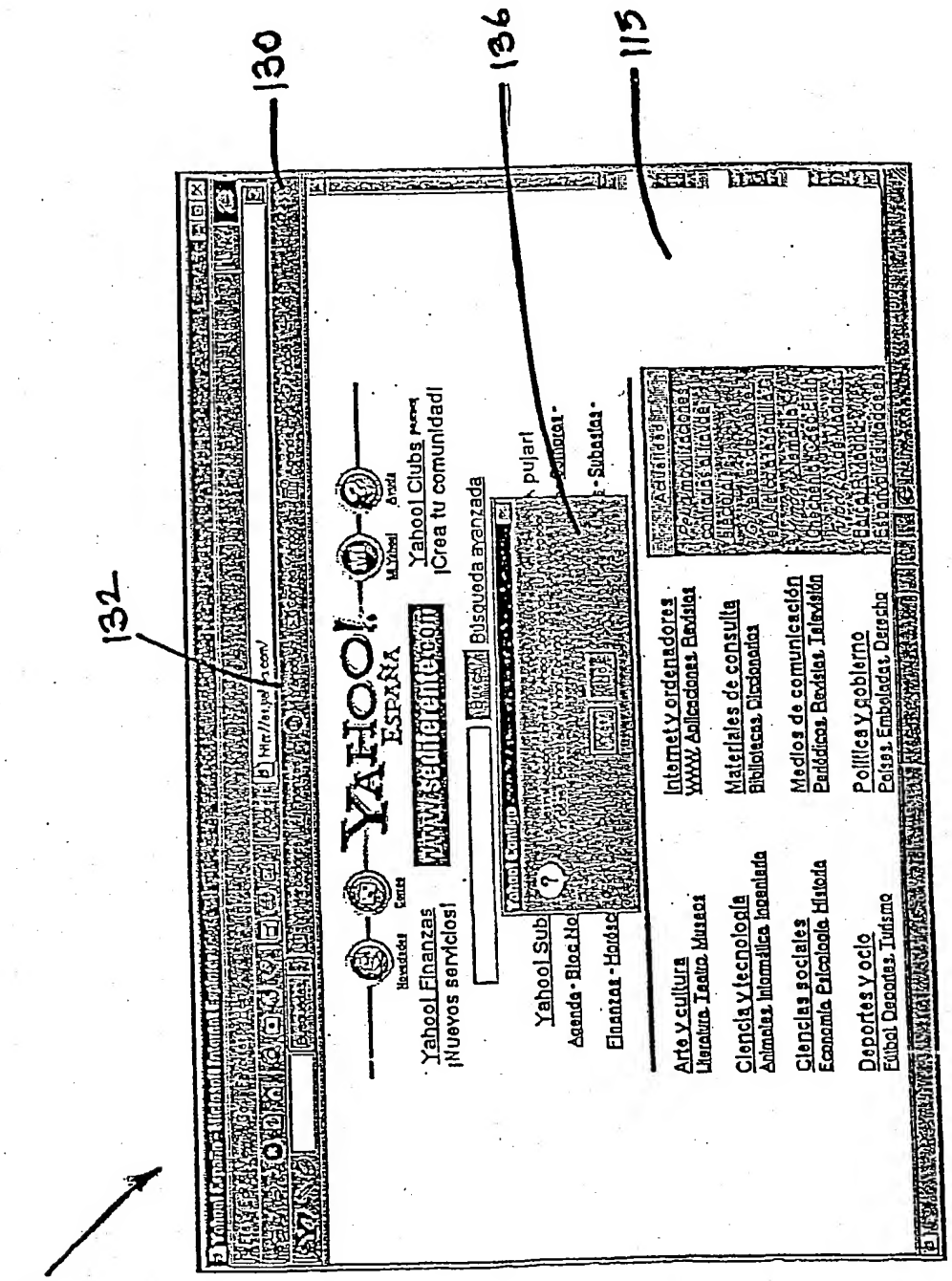

110
132
130
134
115
YAHOO!
ESPAÑA
Yahoo! Finanzas
¡Nuevos servicios!
Búsqueda avanz
Buscar: Todo español Sólo España
Yahoo! Subastas: Vender en Yahoo! Subastas es GRATIS ¡A pujar!
Agenda - Blog Notas - Busca gente - Chats - Clasificados - Clubs - Correo - Comprar - Contactos - Deportes
Finanzas - Horóscopo - MiYahoo! - Noticias - Messenger - MAVIsat - Pasatiempos - Subastas - Tiempo - más...
Arte y cultura
Literatura, Teatro, Museos
Internet y ordenadores
WWW, Aplicaciones, Revistas
Ciencia y tecnología
Animales, Informática, Ingeniería
Materiales de consulta
Bibliotecas, Diccionarios
Ciencias sociales
Economía, Psicología, Historia
Medios de comunicación
Periódicos, Revistas, Televisión
Deportes y ocio
Fútbol, Deportes, Turismo
Política y gobierno
Países, Embajadas, Derecho

FIG. 10


130
132
136
115
110
YAHOO!
ESPAÑA
Yahoo! Finanzas
¡Nuevos servicios!
Yahoo! Clubs
¡Crea tu comunidad!
Búsqueda avanzada
Arte y cultura
Literatura, Teatro, Museos
Internet y ordenadores
WWW, Aplicaciones, Revistas
Ciencia y tecnología
Animales, Informática, Ingeniería
Materiales de consulta
Bibliotecas, Diccionarios
Ciencias sociales
Economía, Psicología, Historia
Medios de comunicación
Periódicos, Revistas, Televisión
Deportes y ocio
Fútbol, Deportes, Turismo
Política y gobierno
Países, Embajadas, Derecho

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/16993

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 9/45, 9/445
US CL :717/2,11

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 717/2,11 ,1,3; 704/8; 709/203,219,220,221,226

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim
Y	US 5,499,335 A (SILVER et al.) 12 March 1996, col. 3 line 26-col. 7 line 41.	1-49
Y	US 6025836 A (McBRIDE) 15 February 2000, the whole document.	1-49
A, P	US 6,188,995 B1(GARST et al.) 13 February 2001, col. 1 line 10-col. 3 line 47.	1-49
Y,P	US 6,216,153 B1 (VORTRIEDE) 10 April 2001, col. 1 line 6 - col. 3 line 44.	1-49

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priori date and not in conflict with the application but cited to understar the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot b considered novel or cannot be considered to involve an inventive st when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot b considered to involve an inventive step when the document combined with one or more other such documents, such combinatio being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 30 SEPTEMBER 2001	Date of mailing of the international search report 25 OCT 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer KAKALI CHAKI Telephone No. (703) 305-9600

Form PCT/ISA/210 (second sheet) (July 1998) ★